Spring Integration

Photo credit: ***Michelangelo***

Today's business applications rarely live in isolation. That's why we have often in applications development to implement a communication module to make the new application exchanging data with the existent information system. To implement those data flow we can use many technologies (JMS, SOA, Apache commons IO ...). Managing all those components in a project can be tricky and not easily maintainable.

Spring Integration provides an extension of the Spring programming model to support the well-known Enterprise Integration Patterns. It enables lightweight messaging

Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use.
To find out more, including how to control cookies, see here: Cookie Policy

Close and accept

The first part of this article presents some basic background to start using Spring Integration. The second part deals with a project sample.

1- Spring Integration Basics

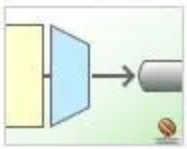
Spring Integration adds essentially three components to the core Spring Framework: messages, message channels, and endpoints.

1.2-Spring Integration components

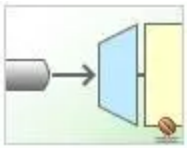
In this part we will present some basic components used in this article sample.

Message: is a data structure that will have payload and the header. The payload can be a file, String, JMS message … The header is a key-value map that contains some properties that will be used by the framework or user values.

Channel: it represents the "pipe" of a pipes-and-filters architecture. Producers send Messages to a channel, and consumers receive Messages from a channel.



Inbound Channel: it does some mapping between the Message and whatever object or resource is received-from the other system (File, HTTP Request, JMS Message, etc).



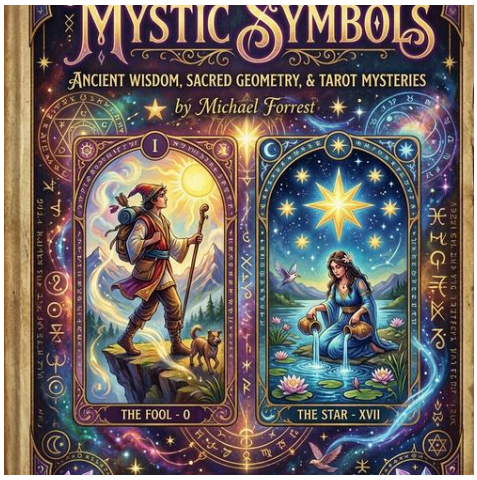
Outbound Channel: it does the same thing like the inbound Channel but in the other side. It map message to send to other system.

Service Activator: it's a generic endpoint for connecting a service instance to the messaging system. The input Message Channel must be configured, and if the service method to be invoked is capable of returning a value, an output Message Channel may also be provided.

Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use.
To find out more, including how to control cookies, see here: Cookie Policy

Transformer: is used to create a Message-transforming endpoint. It requires a "ref". The "ref" may either point to an Object that contains the @Transformer annotation on a single method or it may be combined with an explicit method name value provided via the "method" attribute.[2]

Splitter: is a component whose role is to partition a message in several parts, and send the resulting messages to be processed independently.



Unlock Ancient Wisdom

Explore 52 years of tarot insights and sacred geometry designed to deepen your intuition.

Discover the book

To get more details about those components and to discover others components you can read the Spring integration reference manual in this link : http://static.springsource.org/spring-integration/reference/htmlsingle/

1.2- Spring integration component specification

To specify components, you can use the basic xml language or annotations.

This source code presents the transformer declaration by sing the XML language.

```
1 <integration:transformer input-channel="channel1" output-channel="channel2">
2     <bean class="com.mycompany.integration.FileTransformer" />
3 </integration:transformer>
```

Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use.
To find out more, including how to control cookies, see here: Cookie Policy

The same thing could be done by using Spring integration annotation.

First add to your spring application context the declaration of the package where your annotated classes are defined.

```
<integration:annotation-config/>
<context:component-scan base-package="com.mycompany.integration"/>
```

Next you add the annotation to your transformer class.

```
@Component
public class FileTransformer {

    @Transformer(inputChannel="channel1", outputChannel="channel2")
    public NewMessageFormat transform(OldMessageFormat omf) throws IOException{
        //Your transformation implementation here
        return newMessageFormat;
    }
}
```

I recommend the xml specification because it could be used by STS to display integration graph. This graphic representation is helpful for reading and verifying the integration flow.

1.3- Create Spring integration project

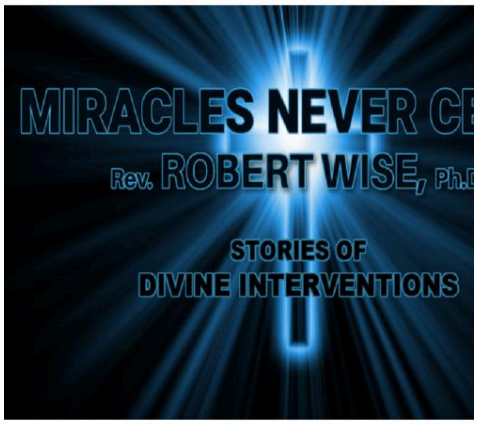
The simplest way to create a Spring Integration project is to create a basic Spring project and to add Spring Integration dependencies to the pom file.

```
<dependency>
    <groupId>org.springframework.integration</groupId>
    <artifactId>spring-integration-core</artifactId>
    <version>${spring.integration.version}</version>
</dependency>

<dependency>
    <groupId>org.springframework.integration</groupId>
    <artifactId>spring-integration-file</artifactId>
    <version>${spring.integration.version}</version>
</dependency>

<dependency>
    <groupId>org.springframework.integration</groupId>
    <artifactId>spring-integration-jdbc</artifactId>
    <version>${spring.integration.version}</version>
</dependency>
```

Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use.
To find out more, including how to control cookies, see here: Cookie Policy



When Shadows Speak

Ever felt a presence no one else hears? Discover what happens when spirits cross into reality.

Read the full story

1.4- Spring Tools Suite (STS)

You can develop your spring integration application with your favorite IDE without any extra feature. But I think it is too much easier if it will be done with STS. You can download STS from this link http://www.springsource.com/developer/sts or to install the STS plugin for eclipse form this link http://marketplace.eclipse.org/content/springsource-tool-suite.

2- Spring integration sample

In this sample, we suggest to implement data integration between two systems. The first one produces data into flat file and the second system receives a data into a database.

You can import the project sample from google code SVN link:
http://project4example4.googlecode.com/svn/trunk/

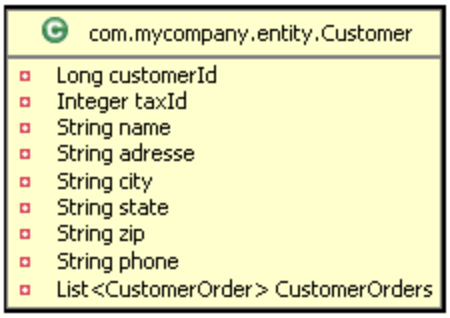
The application needs to detect the new file in a folder, log the name of the detected file, read text file and integrate its content into the database.

Bellow we present the source formatted flat text file content and the destination persistent classe.

Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use.
To find out more, including how to control cookies, see here: Cookie Policy

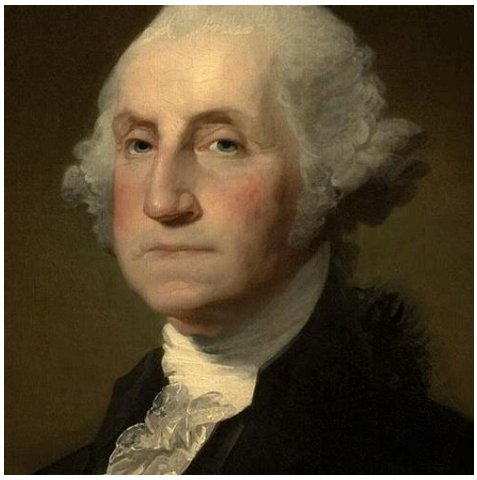
```
id tax name adresse city state zip phone
5 2 name5 adresse5 Paris Paris 75001
0606060606
6 2 name6 adresse6 Paris Paris 75001
0606060606
7 2 name7 adresse7 Paris Paris 75001
0606060606
8 2 name8 adresse8 Paris Paris 75001
0606060606
9 2 name9 adresse9 Paris Paris 75001
0606060606
```

Source file format



Persistent class

The picture bellow presents Spring integration graph used to implement the solution.

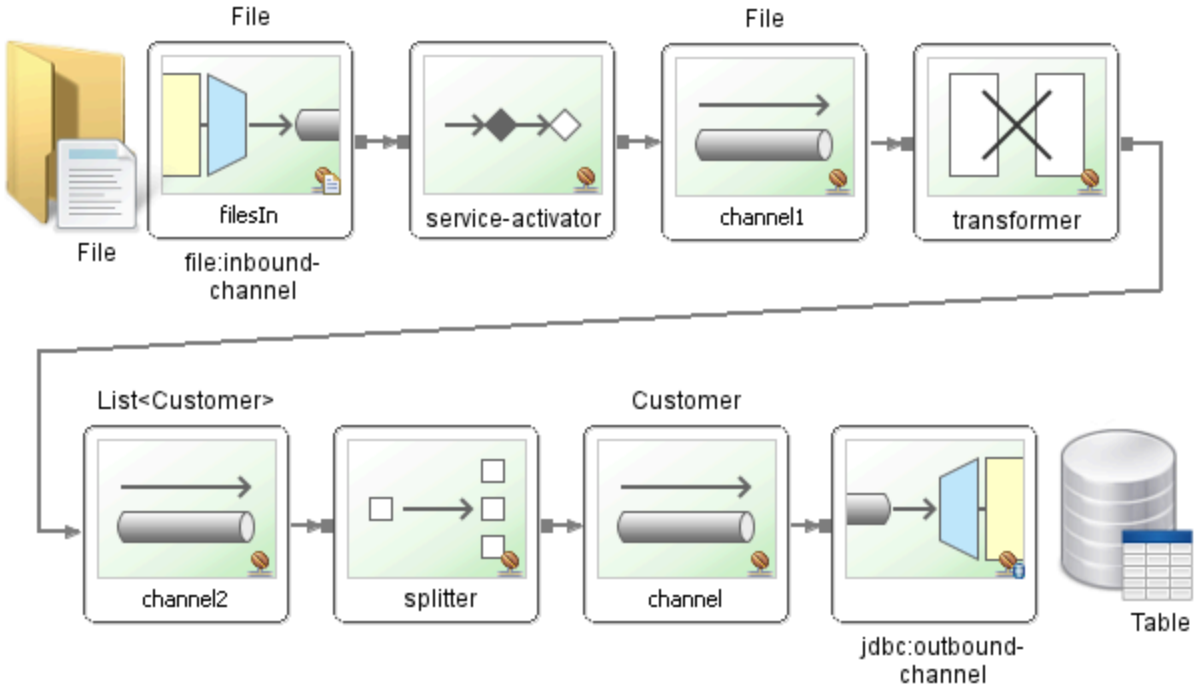


Washington: Free Speech Or Else

If free speech is punished, reason fails and we risk becoming like silent sheep led to the ...

Read Washington's Speech

Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use.
To find out more, including how to control cookies, see here: Cookie Policy



The file inbound channel inspect every 5 second the input folder. If a new file is present a message with a payload of type file will be created and sent to the service activator.

The service activator in this sample will just log the file name and deliver the message to the channel1.

The Transformer gets the Message from channel1, parse it, create a list of Customers and send the new message content to channel2.

The splitter gets the message, read the payload content and split the message to massages with a payload of type Customer.

The JDBC outbound channel gets the payload and uses payload attribute to execute the query.

The code source bellow presents the xml code of the spring integration graph. You can edit your Spring integration flow with both modes graphic and XML code mode. If you change XML code, the graph will be updated and if you use the graphic mode the changes done to the graph will affect the source code.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans xmlns="http://www.springframework.org/schema/beans"
3    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
8  http://www.springframework.org/schema/beans/spring-beans.xsd
```

Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use.
To find out more, including how to control cookies, see here: Cookie Policy

```
http://www.springframework.org/schema/integration
http://www.springframework.org/schema/integration/spring-integration-2.0.xsd
http://www.springframework.org/schema/integration/file
http://www.springframework.org/schema/integration/file/spring-integration-file.xsd
http://www.springframework.org/schema/integration/jdbc
http://www.springframework.org/schema/integration/jdbc/spring-integration-jdbc.xsd">

<bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer" />

    <file:inbound-channel-adapter id="filesIn"
        directory="${customer.file.input.path}">
        <integration:poller id="poller" fixed-delay="5000"/>
    </file:inbound-channel-adapter>

    <integration:service-activator   input-channel="filesIn" ref="handler" output-channel="c

    <bean id="handler" class="com.mycompany.integration.Handler" />

    <integration:channel id="channel1" />

    <integration:transformer input-channel="channel1" output-channel="channel2">
        <bean class="com.mycompany.integration.FileTransformer" />
    </integration:transformer>


    <integration:channel id="channel2" />

    <integration:splitter expression="payload" input-channel="channel2"  output-channel="dbC

    <integration:channel id="dbChannel" />

    <jdbc:outbound-channel-adapter
        query="insert into customer (customer_id, tax_id, name, adresse, city, state, zip, p
        (:payload.customerId, :payload.taxId, :payload.name, :payload.adresse, :payload.city
        channel="dbChannel" data-source="dataSource" />
```

To test the application, you can run the jUnit test “testIntegrateCustomer”.

```
@Test
    public void testIntegrateCustomer() throws Exception {

        Collection<Customer> listCustomers = iCustomerDao.getAll();
        Assert.assertEquals(3, listCustomers.size());
        logger.info("customers number : {}", listCustomers.size());

        logger.info("wait .... ");
        Thread.sleep(2000);

        URL customerFileUrl = this.getClass().getResource("/input/customer.txt");
        File customerFile = new File(customerFileUrl.toURI());
        Assert.assertTrue(customerFile.exists());


        FileUtils.copyFileToDirectory(customerFile, new File(inputDirectoryPath));
        logger.info("test file {} copied to {}", customerFile.getName(), inputDirectoryPath)
```

Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use.
To find out more, including how to control cookies, see here: Cookie Policy

```
        listCustomers = iCustomerDao.getAll();
```

```
23          logger.info("new customers number : {}", listCustomers.size());
24          Assert.assertEquals(8, listCustomers.size());
25
```

Spring Integration is a smart solution for application data integration. It provides support for the standard patterns typically used in enterprise integration. Spring Integration adds lightweight messaging and support for integrating with external systems and services using an adapter framework. Adapters enable a higher-level abstraction for communication with external systems and for Spring's support for data and remote access as well as scheduling.

Reference

[1] http://www.eaipatterns.com/

[2] http://static.springsource.org/spring-integration/reference/htmlsingle/

[3] Dr. Mark Lui, Mario Gray, Andy H. Chan, and Josh Long, Pro. Spring Integration. Apress.

Posted August 1, 2011 in Java
by Samer Abdelkafi

Tags:

Database, ESB, Framework, Spring, Spring Integration, Web service

Comments

Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use.
To find out more, including how to control cookies, see here: Cookie Policy

Samer Abdelkafi Blog

Create a free website or blog at WordPress.com.

Privacy & Cookies: This site uses cookies. By continuing to use this website, you agree to their use.
To find out more, including how to control cookies, see here: Cookie Policy